E.02.14 – D3.5 – CASSIOPEIA – User Manual

Document information	
Project Title	E.02.14-D3.5-CASSIOPEIA-User Manual
Project Number	E.02.14
Project Manager	The Innaxis Foundation and Research Institute
Deliverable Name	CASSIOPEIA User Manual
Deliverable ID	D 3.5
Edition	00.00.01
Template Version	03.00.00
Task contributors	
Universidad Politécnica de Madrid	

Abstract

The goal of Cassiopeia's WP3 is the development of a software system for agent-based simulation in the ATM domain. This document is the user manual of the implemented software system and it has been written as a result of the activity developed in WP3.4 "Software Programming". The document explains how to install and use the software system, describing how to prepare and execute a simulation case and how to consult the generated results.

Authoring & Approval

Prepared By - *Authors of the document.*		
Name & Company	Position & Title	Date
Jorge Martin / Universidad Politécnica de Madrid	Consortium Member	08/03/2013
Sergio Carrasco / Universidad Politécnica de Madrid	Consortium Member	08/03/2013
Natalia Stulova / Universidad Politécnica de Madrid	Consortium Member	08/03/2013
Martin Molina / Universidad Politécnica de Madrid	Consortium Member	08/03/2013

Reviewed By - *Reviewers internal to the project.*		
Name & Company	Position & Title	Date
Samuel Cristóbal Centenera	Consortium Member	12/03/2013

Reviewed By - *Other SESAR projects, Airspace Users, staff association, military, Industrial Support, other organisations.*		
Name & Company	Position & Title	Date
-	-	-

Approved for submission to the SJU By - *Representatives of the company involved in the project.*		
Name & Company	Position & Title	Date
David Perez / The Innaxis Foundation and Research Institute	Consortium Coordinator	18/03/2013

Rejected By - *Representatives of the company involved in the project.*		
Name & Company	Position & Title	Date
-	-	-

Rational for rejection
N/A.

Document History

Edition	Date	Status	Author	Justification
00.00.01	01/02/2013	Draft	UPM	New Document

Intellectual Property Rights (foreground)

This deliverable consists of Foreground owned by one or several Members or their Affiliates.

Table of Contents

List of figures:

1 Introduction

1.1 Purpose of the Document

This document is the user manual of the software system for agent-based simulation developed in the CASSIOPEIA project. The goal of the software system is to assess different potential changes of ATM strategies and the resulting impact on air traffic operations.

The purpose of this document is to explain how to use the software system. In particular, the document describes (1) how to install the software system, describing software and hardware requirements, (2) how to prepare a case for simulation, explaining how to configure ATM networks and the parameters of regulation policies, and finally (3) how to consult and visualize the generated results.

This deliverable takes input mainly from the following deliverables related to WP3: D3.1 Software Requirements, D3.2 Software Design, D3.4 System Implementation and D3.6 System Evaluation.

1.2 Intended Readership

The document is oriented to readers interested in using the CASSIOPEIA software system to execute simulations. The document describes technical details about the installation, configuration and execution of the software system. It is assumed that the readers are familiar with installation and execution procedures in Windows operating system (Windows 7) and they are familiar with agent-based configuration using declarative languages (e.g., XML) and other software tools (relational databases and visualization tools).

1.3 Structure of the document

This document is structured as follows:

- Chapter 1 covers the introduction of this document.
- Chapter 2 describes the installation procedure together with hardware and software requirements.
- Chapter 3 explains how to prepare a case for simulation with the software platform describing the definition of ATM networks and the regulation policies.
- Chapter 4 explains how to execute a simulation and how to consult and visualize the generated results.

1.4 Acronyms and Terminology

Term	Definition
ADF	XML language subset for describing agents
AMS	Agent Management System (i.e., a specialized agent). It represents the authority in the platform. It is the only agent that can create and kill other agents, kill containers, and shut down the platform.
AOSE	Agent Oriented Software Engineering
Architecture	The structure or structures of the system. They comprise software components, the externally visible properties of those components, and the relationships among them.
ATM	Air Traffic Management
ATMS	Air Traffic Management System
BDI	Belief desire intention model

Class	It is a construct that is used to create instances of itself – referred to as class instances, class objects, instance objects or simply objects. A class defines constituent members which enable its instances to have state and behavior. Data field members (member variables or instance variables) enable a class instance to maintain state. Other kinds of members, especially methods, enable the behavior of class instances. Classes define the type of their instances.
Component	A component is a modular part of a system. It is a re-usable piece of software that has a well specified public interface and implements a limited functionality.
DDR	Double Data Rate
CSV	Comma separated values
DF	Director Facilitator (i.e., an specialized agent). It implements a yellow pages service which advertises the services of agents in the platform so other agents requiring those services can find them.
FIPA	The Foundation for Intelligent Physical Agents
Framework	Represents a collection of conceptual and technological mechanisms that provide a set of services and guidelines for applying a problem to a particular domain. It aims to support and help resolve specific, actual domain problems or similar theoretical problems. In terms of software, it provides some classes that clients can use or adapt. A framework realizes an architecture.
HDD	Hard disk drive
IATA	International air transport association
ICAO	International civil aviation organization
JADEX	A software framework for the creation of goal-oriented agents following the belief-desire-intention (BDI) model.
JDK	Java development kit
KPI	Key Performance Indicator
JVM	Java virtual machine
JAR	Java Archive
RDBMS	Relational database management system
SESAR	Single European Sky ATM Research Programme
SESAR Programme	The programme which defines the research and development activities and projects for the SJU.
SJU	SESAR Joint Undertaking (Agency of the European Commission)
SQL	Structured query language
SW	Software
UML	Unified Modeling Language
XML	eXtensible Markup Language

2 System installation

This chapter explains how to install the CASSIOPEIA software system, describing the software and hardware requirements together with the installation procedure.

2.1 Software requirements

The CASSIOPEIA software system requires the following software tools for a correct execution:

- *Operating system*. The system operates in Microsoft Windows as operating system (Windows 7).
- *Java virtual machine*. The system requires a Java virtual machine (JVM) for the execution. In particular, JVM SE 7 is the selected virtual machine for the Cassiopeia software tool [Zakhour et al., 2013].
- *Data base manager*. The system uses MySQL database server (version 5.5) for data storage [MySQL AB, 2006].This also includes a database administrator tool MySQL Workbench.

The system uses the following software libraries:

- The library JADEX 2.2.1 [Pokahr, 2012] provides interaction functionalities between agents and agent reasoning with XML configurations.
- The library Jcoord 1.0 [Stott, 2006] provides some functions to calculate distances between geographical points.
- The library MySQL-connector-java 5.1.22 is a library to handle connections between a MySQL database and a java implementation [MySQL AB, 2006].

2.2 Hardware requirements

The CASSIOPEIA software system operates in general purpose computers with Windows 7 operating system. The minimum hardware requirements for the system execution are the following:

- Processor: Intel Core i5-2250
- RAM : 8 GB DDR3
- Storage: HDD 1000 GB 7200rpm

The following hardware requirements are recommended for a more efficient execution of simulations:

- Processor: AMD Hexa-core, 6 cores x 2,8 GHz (3,3 Turbo Core)
- RAM: 16 GB DDR3 ECC
- Storage: RAID 1 : 2x HDD 1000GB 10000rpm

2.3 Installation procedure

The software system is delivered in the form of three computer files:

- File *dump.sql*: SQL dump file for creation of a database structure contained in a SQL file.
- File *simulation-platform.zip*: Simulation platform contained in a zip file.
- File *vizCassiopeia.zip*: Visualization tool contained in a zip file.

The installation procedure covers three main steps: (1) initialize the database, (2) install the simulation platform (3) install the visualization tool. The following paragraphs describe these steps:

- *STEP 1: Initialize the database*

  The database initialization includes the following steps to import the provided SQL dump file into the RDBMS:

  1. The user initiates the execution of MySQL Workbench. The user connects the Cassiopeia database using the server administration section in the MySQL Workbench (Figure 2.1).
  2. The Workbench displays the server administration window (Figure 2.2). The user chooses the option *data import task*. Then, the user selects the dump file (file *exports.sql* in the figure) and clicks the button *start import*.

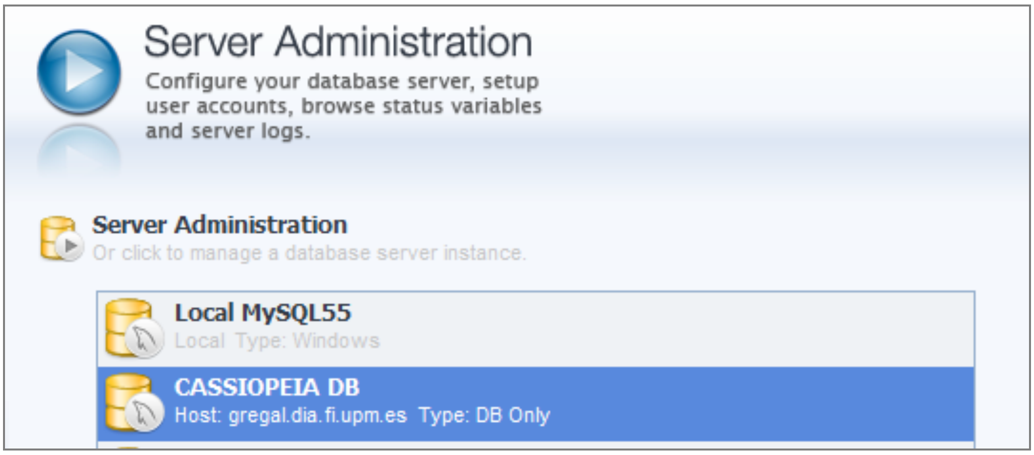

Figure 2.1: Server administration section in the MySQL Workbench

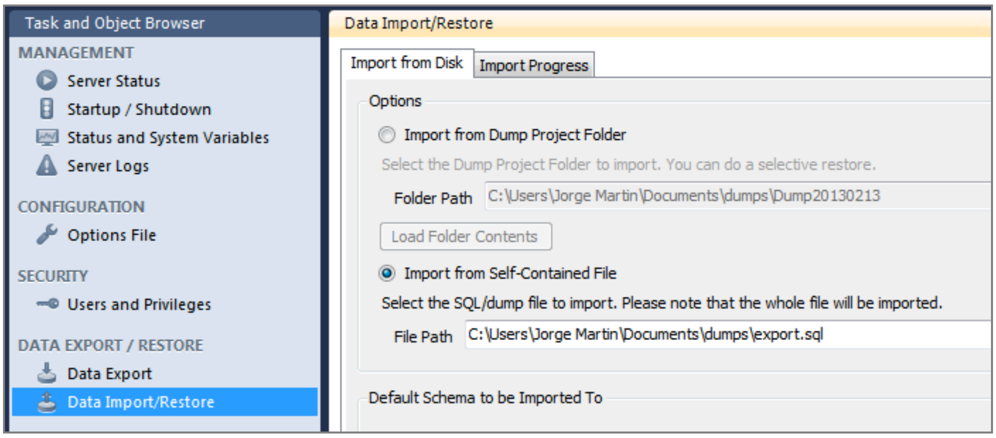

Figure 2.2: Server administration window

- *STEP 2: Install the simulation platform.*

  To install the simulation platform, the user extracts the contents of the zip file into a folder (Figure 2.3).

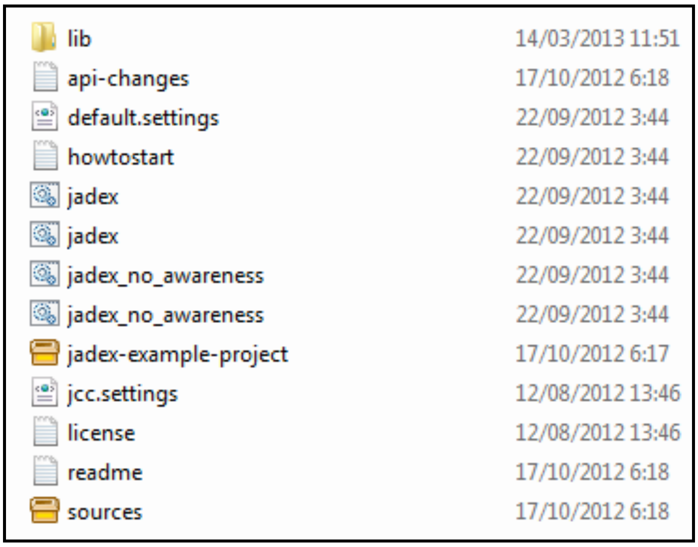

Figure 2.3: Content of the folder simulation-platform

- *STEP 3: Install the visualization tool.*

  To install the visualization tool, the user extracts the contents of the zip file into a folder (figure 2.4).

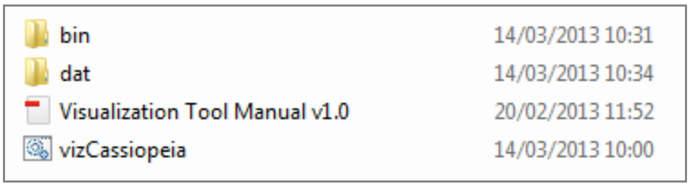

Figure 2.4: Content of the folder vizCassiopeia

3 Preparation of a simulation case

This section describes how to prepare a case for simulation with the CASSIOPEIA software platform. The preparation of a simulation case for execution requires performing two tasks: (1) definition of the ATM network and (2) definition of a regulation policy. The following sections describe in more detail these tasks. Appendix A shows an example of how to define the ATM network with illustrative formal models (XML files and database tables).

3.1 Definition of the ATM network

The goal of this task is to formulate the details of a specific ATM network for simulation. The network is represented using a flexible agent-based approach describing both the structure and their operational behavior. This representation includes, for example, specific airports, airlines and flight plans. In the CASSIOPEIA project, ATM networks for three different case studies have been defined, although the platform has been designed as a general solution to accept other cases. The definition of the network is performed in two steps: (1) definition of the case-specific agent model and (2) definition of the specific components of the ATM network (agent instances and environment).

3.1.1 Case-specific agent model

The definition of an ATM network includes the specification of general behavior and properties that are common for specific agent instances. This includes the definition of a case-specific agent model with the specification of the functional description of agents in terms of capabilities. This description specifies, for example, that an airport is able of requesting new flight schedules and selecting the best schedule option.

```
<plans>
  <plan name="applyRegulation" >
    <body class="ApplyRegulationPlan"/>
    <trigger>
      <messageevent ref="inform_regulation"/>
    </trigger>
  </plan>
  <plan name="slotAssignmentPlan" >
    <body class="SlotAssignmentPlan"/>
    <trigger>
      <internalevent ref="perform_assignment" />
    </trigger>
  </plan>
  </plan>
  <plan name="rescheduleConfirmation">
    <body class="RescheduleConfirmationPlan"/>
    <trigger>
      <messageevent ref="schedule_confirmation"/>
    </trigger>
  </plan>
</plans>
```

Figure 3.1: Partial example of the definition of agent-plans for an airport (XML language)

The software platform provides reusable definitions for ATM agents and their capabilities (e.g., airport, airlines, etc.), but new capabilities for a specific ATM network of a case study can be formulated using a flexible agent-based approach using XML language with beliefs, goals and plans (see figure 3.1). Appendix A illustrates with a complete example how to define a case-specific agent model with three types of XML files: (1) agent capability (file extension *.capability.xml*), (2) the manager agent (file *manager.agent.xml*) and (3) the application file (file extension *.application.xml*).

The simulation definition for a particular ATM network can include also case-dependent algorithms that simulate specific agent plans. For example, the agent airport can include a plan called *slotAssignementPlan* (Figure 3.1) with a specific algorithm that simulates how an airport tries to re-assign slots to consider new constraints. The software platform provides

general algorithms for agent plans, but new algorithms for agent plans can be added for a specific ATM network of a case study. This definition can be formulated in Java language as an extension of generic plans in Jadex (see details about this definition in D.3.4 System Implementation).

The set of files corresponding to the case-specific agent model are packed into a JAR file with the name of the case (for example, *case01.jar*) to be used by the simulation engine (see section 4.1). This file can be stored in a specific folder defined by the user with the JAR files of other simulation cases.

3.1.2 Agent instances and environment

The definition of the ATM network also includes the specification of particular agent instances and the agent-environment. This corresponds to the definition of all the airports, airlines and flight plans considered in the simulation case. The software platform allows a flexible definition of this information by using a relational database, with the facilities provided by a database management system to load the data from different information sources. With this solution, it is possible to simulate easily different ATM networks for the same case study with different geographic areas and/or different sizes, and compare the impact of the same regulations.

idagent	idclass	name	code	location	latitude	longitude
1	3	Madrid Barajas	LEMD	Madrid	40.4500000	-3.5500000
2	3	Barcelona El Prat	LEBL	Barcelona	41.2969400	2.0783300
3	3	Athens International Airport	LGAV	Atenas	37.9363889	23.9444444
4	4	Warsaw Modlin Airport	EPMO	Nowy Dwór Mazowiecki	52.4511110	20.6516670
5	4	Valladolid	LEVD	Valladolid	41.7061120	-4.8519440
6	4	Girona	LEGE	Girona	41.9008330	2.7605560
7	5	Iberia	IBE	Madrid	40.4448570	-3.6842540
8	5	Egyptair	MSR	Estambul	41.0638220	28.9928630
9	5	Air Europa	AEA	Mallorca	39.4833330	2.8511110
10	5	Air France	AFR	París	49.0097220	2.5477780
11	5	Air Nostrum	ANE	Valencia	39.4666667	-0.3833333
12	5	Alitalia	AZA	Fiumicino	41.7666670	12.2333330
13	5	Eurotrans	BCS	Zaventem	50.8867920	4.4748120
14	5	British Airways	BAW	Harmondsworth	51.4865000	-0.4796000
15	5	Brussels Airlines	BEL	Diegem	50.8942320	4.4356230
16	5	Czech Airlines	CSA	Prague	50.0833330	14.4166670
17	5	Lufthansa	DLH	Cologne	50.9500000	6.9666670
18	6	Easyjet	EZY	Luton	51.8747222	-0.3683333
19	5	FlightLine	FLT	Rochford	51.5821000	0.7065000
20	5	Spanair	JKK	L'Hospitalet de Llobre...	41.3588890	2.0994170
21	5	Blue Air	JOR	Bacău	46.5219440	26.9102780
22	5	KLM Royal Dutch Airline	KLM	Amstelveen	52.3000000	4.8700000
23	5	Privilege Style	PVG	Palma, Majorca	39.5666670	2.6500000
24	6	Ryanair	RYR	Leinster	53.3477780	-6.2597220
25	5	Solinair	SOP	Sečovlje	45.4755030	13.6168670
26	5	Swift Air	SWT	Madrid	40.4000000	-3.6833330

Figure 3.2: Window of the MySQL workbench user interface

The implementation of the relational database uses the MySQL workbench, with a user-friendly interface (Figure 3.2), that helps the user to easily load and manipulate the content of the database. For example, the user can load data automatically from external CSV files according to the structure of tables. The definition of agent instances and the environment uses a set of database tables provided by the CASSIOPEIA software platform (Figure 3.3). The steps to create a case-specific content of the database are the following (Appendix A illustrates this definition with examples for each table):

- *Create agent instances*. To create agent instances, a new record is created in the *AGENT* table for each new agent. Before the agent creation, defining the agent classes in the *AGENT_CLASS* table is required. The particular properties of the agents are created in the *AGENT_ATTRIBUTE* table.
- *Create the flight plan*. The flight plan is added to the *FLIGHT* table. A new record is created for each flight in the flight plan. The *AIRCRAFT* table should include the previously required data related to flights.

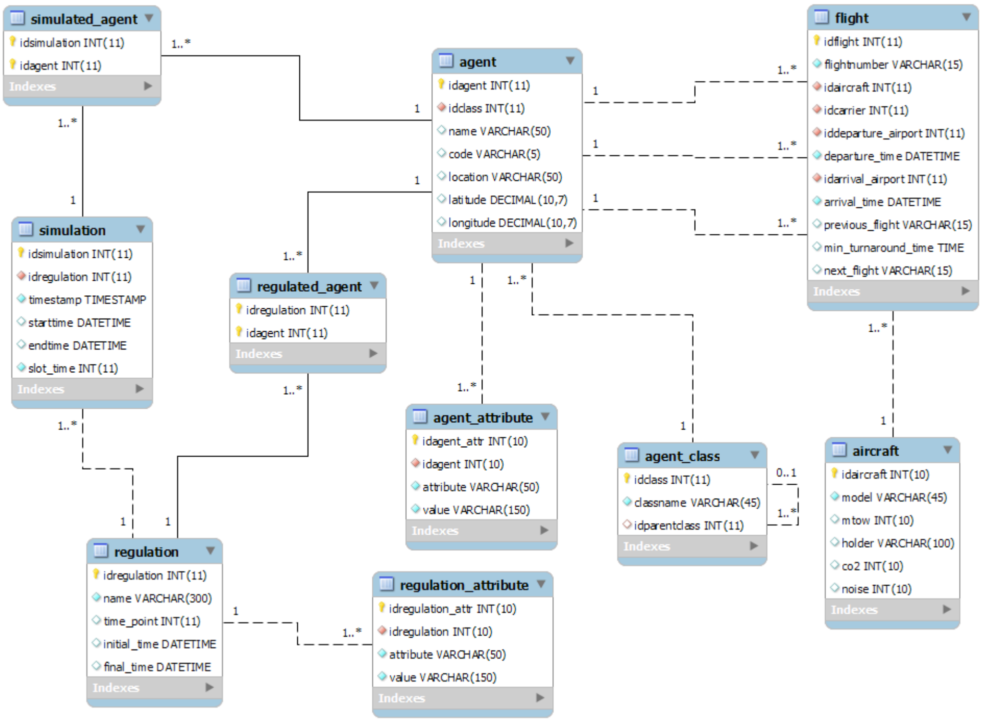

Figure 3.3: Tables of the database related to agent instances and environment

3.2 Definition of a regulation policy

The definition of a regulation policy specifies a hypothesis of regulation to be simulated with an ATM network. Different regulation policies can be simulated for the same virtual ATM network, so that the software platform can help the user to understand the impact of changes in regulation policies. A regulation policy may be, for example, considering new time constraints for flights at certain airports. It is assumed that a regulation policy can be formulated as a set of parameters. For example, a hypothesis of regulation is <*Time point* = *T*1*, Start* = 23:00*, End* = 00:00*, Regulated* = [*LEMD*]> with the following parameters:

- *Time point*: The step when the regulation is applied.
- *Start*: Regulation starting time
- *End*: Regulation ending time.
- *Regulated*: List of airports (using the ICAO name).

A Java class called *Regulation* is used to specify the set of parameter values for a regulation hypothesis. The attributes of such a class correspond to the set of parameters and their type of values can be defined according to types provided by Java libraries (e.g., string, integer, date, list, etc.). This declarative solution provides an adequate degree of flexibility and easy integration in the software platform. Figure 3.4 shows the structure of such class. The class also includes functions such as getters, setters and constructors for the automatic manipulation of the attribute values.

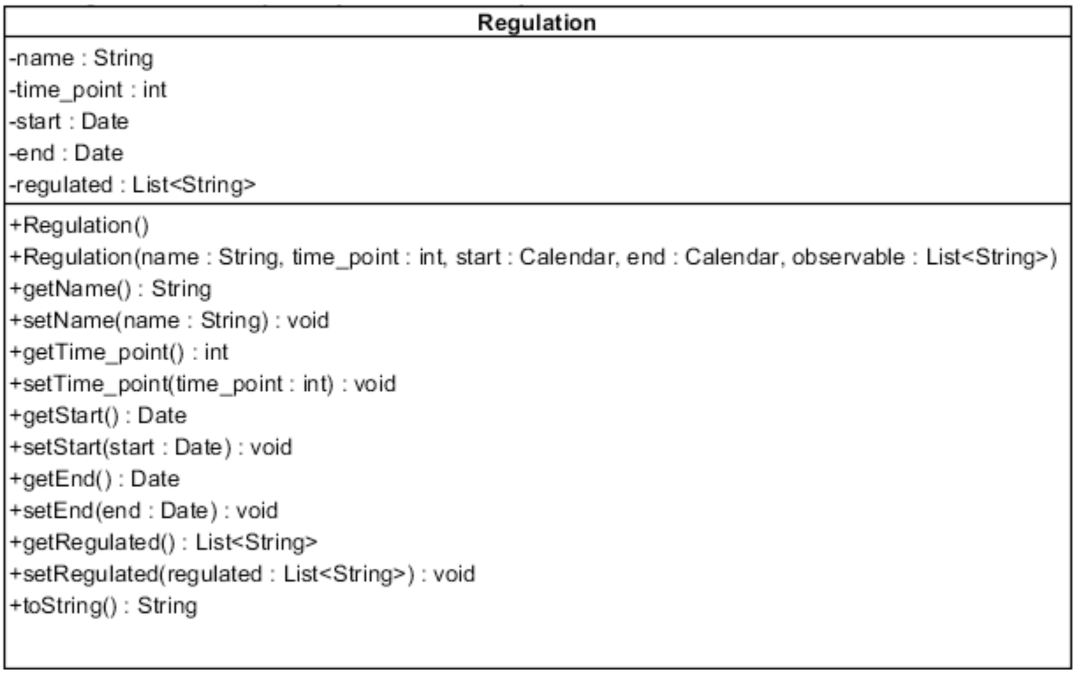

Figure 3.4: Example of class for simulation parameters

4 Simulation execution and analysis of results

This section explains how to execute a particular simulation case and how to consult and analyze the results of the simulation. In particular, there are three methods to analyze to the generated results:

- *Visualization tool.* The visualization tool is used to consult the agent structure and the message interaction.
- *Working memory user interface*. A database user interface is used to consult the specific values of the working memory which contains the initial, intermediate and final generated results.
- *Simulation execution log*. A log text file is generated with a simulation trace describing the linear sequence of steps performed during the execution of the simulation case.

4.1 Control of the simulation execution

To initiate the simulation, the user starts the JADEX control center, by executing the file jadex.bat in the jadex-platform folder (Figure 4.1). Then, the user selects the simulation case by doing the following actions (Figure 4.2):

1. Go to the simulation main window and click on the *add path* icon.
2. Select the file with JAR extension that corresponds to the simulation case and click on *add path*.

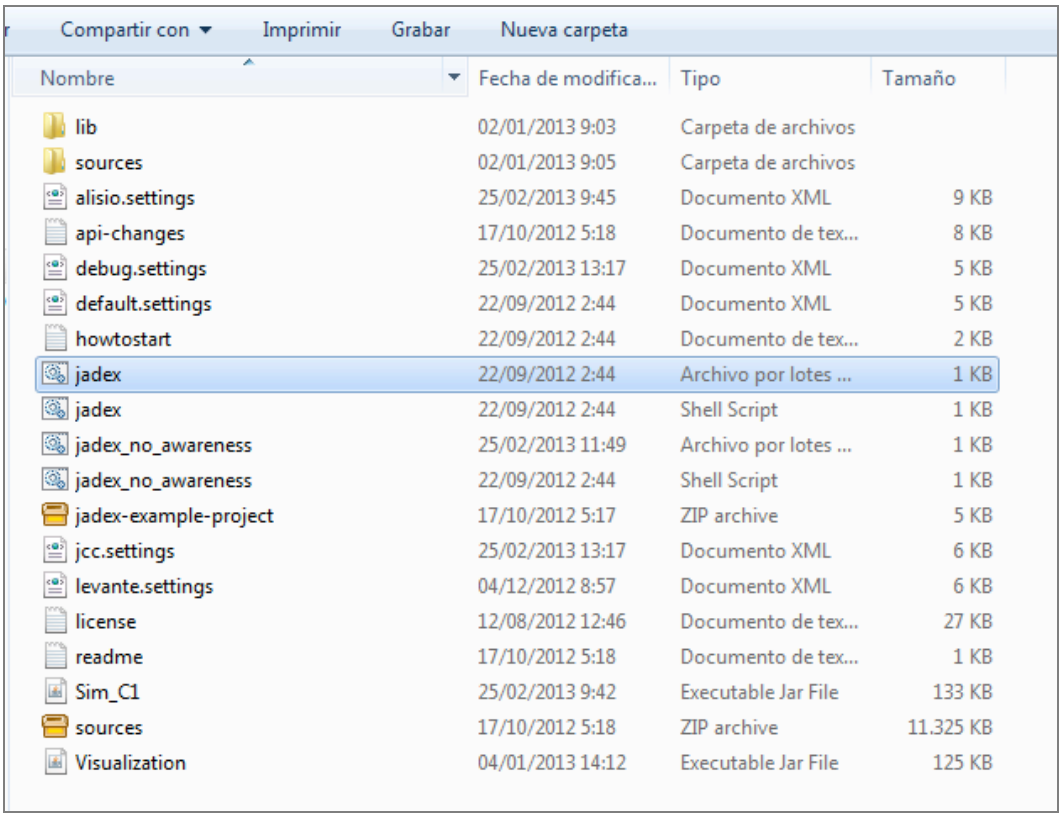

Figure 4.1: JADEX execution

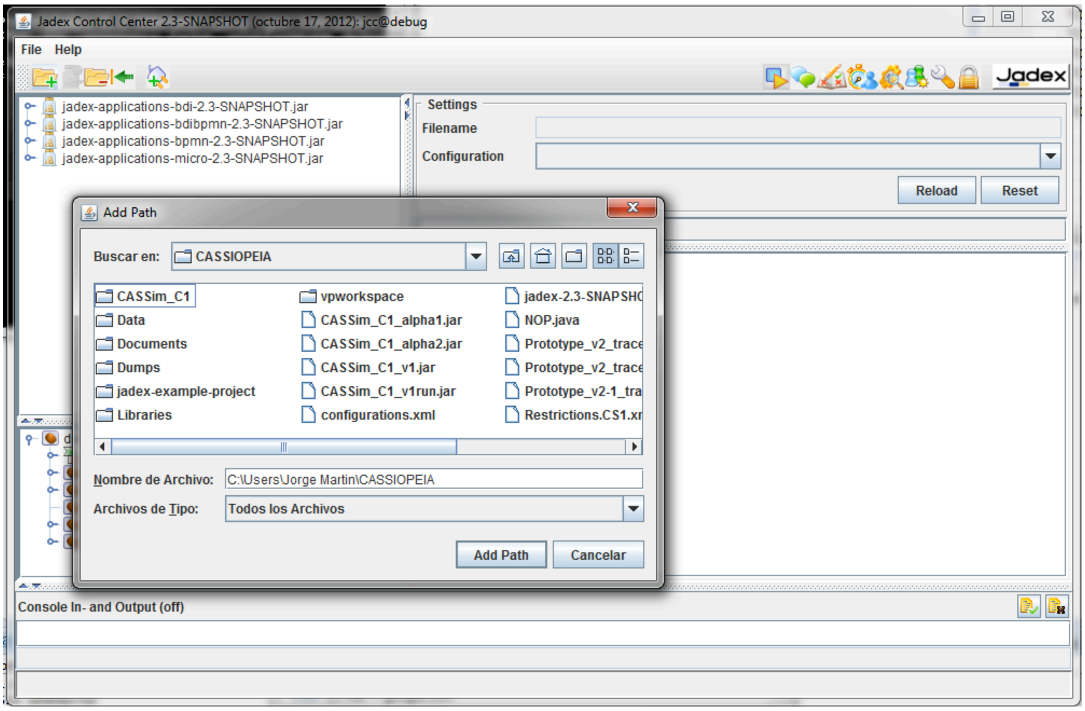

Figure 4.2: Selecting the JAR file corresponding to a simulation case

The user can click on the name of JAR file to expand its content. Then, the user selects the application file corresponding to this case (*C1.application.xml* in Figure 4.3). Finally, the user clicks on the start button to initiate the simulation.

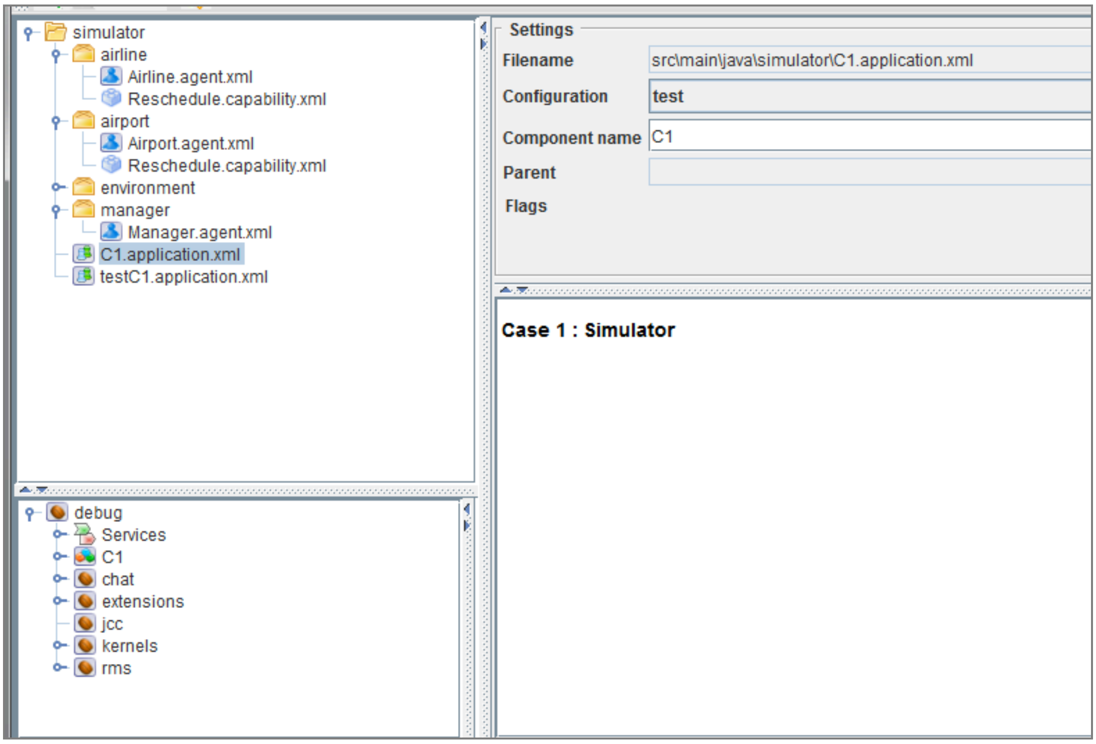

Figure 4.3: Selection of the application file

The simulation window displays logging messages at the bottom of the screen. The user can activate or deactivate logging messages using the buttons next to the top-right corner of the

console. The console notifies that the simulation is finished by displaying a message. Then, the user can close the simulation window and visualize the results using other software tools (e.g., the visualization tool).

The control center provides simulation controls to stop and resume the simulation using the simulation window. The user can access to that window clicking on the simulation icon (figure 4.4). The simulated control window (Figure 4.5) displays a list of execution messages on the right hand side and the controls of the simulation on the left. The user can pause or resume the simulation and also run the simulation step by step.

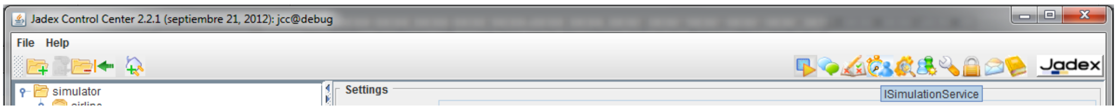

Figure 4.4: Simulation service icon

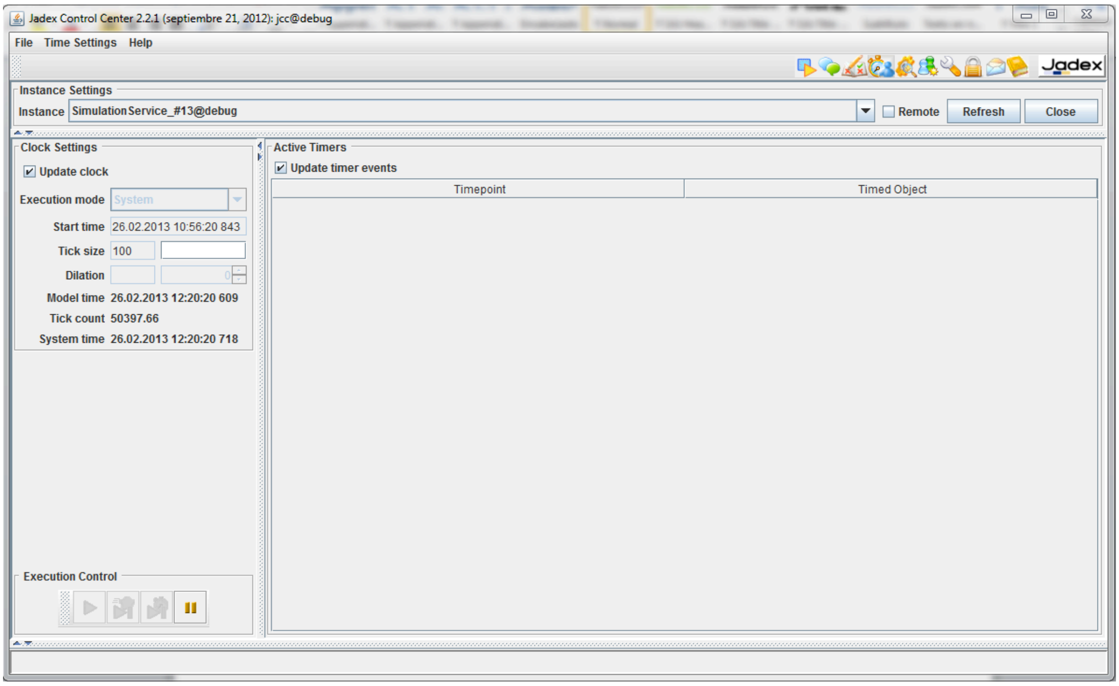

Figure 4.5: Simulation control window

4.2 Visualization tool

A visualization tool can be used to consult the structure of the agent-based model and the agent interaction during the simulation execution. The user starts the visualization tool clicking on the file vizCassiopeia stored in the visualization tool folder (Figure 4.6).

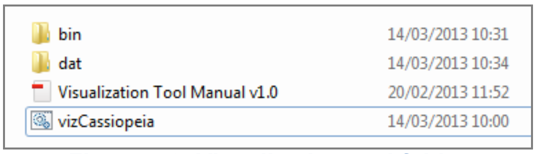

Figure 4.6: Visualization tool folder

The visualization tool starts with an initial window (Figure 4.7). The user can select the simulation ID from the list of available IDs (each simulation ID is automatically assigned by the simulation engine during the execution) and press the button “start visualization”. By default, the first simulation is selected.

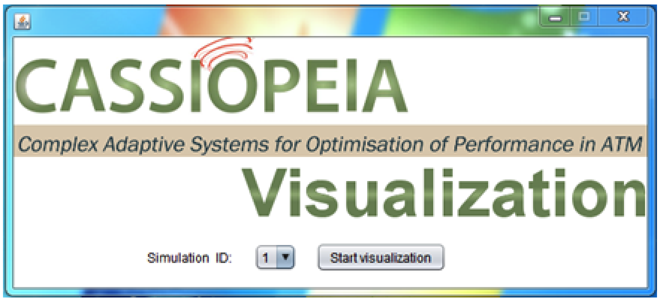

Figure 4.7: Initial window of the visualization tool

4.2.1 Hierarchy view

The hierarchical view is presented after the visualization window (Figure 4.8). This view has two main parts: (1) an interactive tree on the left hand side and (2) a geographic map. The interactive tree presents a structural view of the hierarchical relations between agents. It has two types of nodes: non-leaf nodes that represent categories of agents, and leaf nodes that represent agent instances. By default, all the non-leaf nodes are shown expanded. Figure 4.9 shows the structure of a hierarchy tree corresponding to a simulation case and figure 4.10 shows a fragment of the tree in more detail.

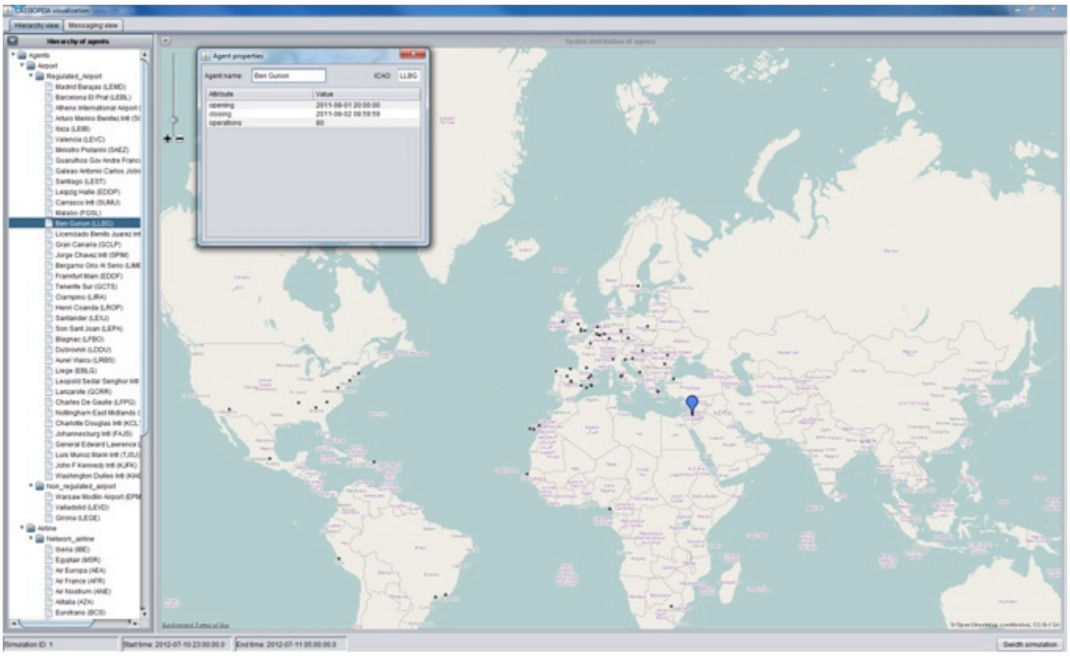

Figure 4.8: Hierarchical view

Level number	Node type	Level name	Entities examples
1	non-leaf	Agent class	Airport, airline.
2	non-leaf	Agent subclass	Regulated airport, network airline, low cost airline.
3	leaf	Agent instance	Madrid-Barajas (LEMD), Iberia (IBE).

Figure 4.9: Example of structure of the hierarchy tree

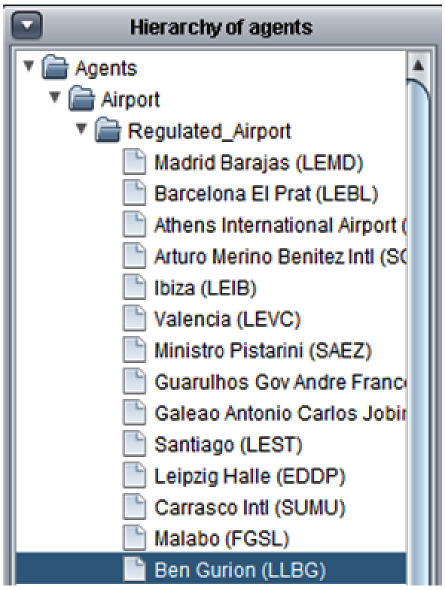

Figure 4.10: A fragment of agent hierarchical tree

Agent instances correspond to the leaf nodes of level 3 of the hierarchy. Each entity is described with the name of the agent and its ICAO code in brackets. By a mouse left-click on a leaf node of a hierarchy two actions take place:

- A map marker, that corresponds to the location coordinates of the agent, is placed on the geographic map,
- A window with properties of the selected agent is shown.

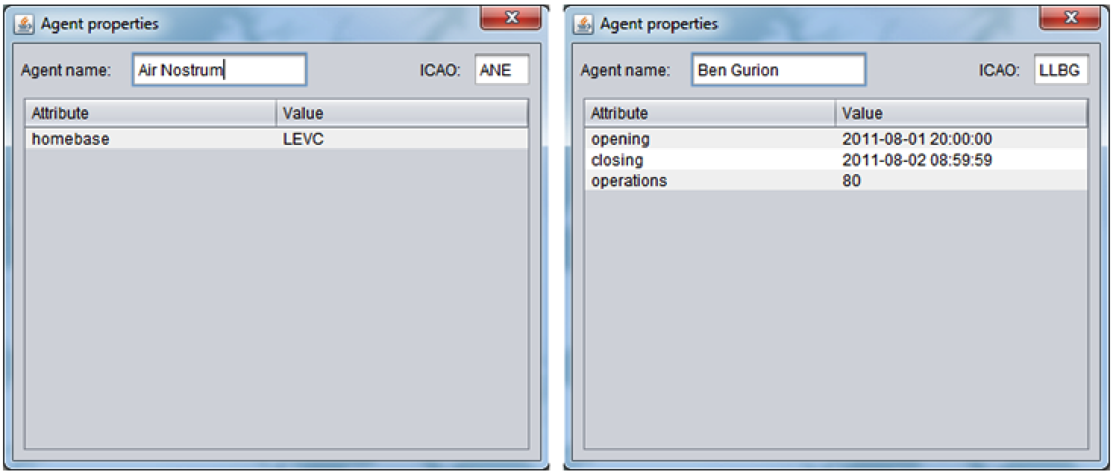

Figure 4.11: Examples of attribute windows for airline and airport agents

The geographic map presents the spatial distribution of the agents. Each agent is represented as a point, which corresponds to its spatial coordinates. For airport agents, spatial coordinates correspond to the airport's location. For airlines, spatial coordinates are the headquarters of the airline. For example, Iberia airline is represented as a point with geographic coordinates near the centre of Madrid City. The map includes standard controls for zooming in and out in the top left corner of it. The user can move the map by clicking on it with the right mouse button and holding it.

4.2.2 Messaging view

Figure 4.12 shows an example screen of the messaging view provided by the visualization tool. This screen has two main sections: “messages” and “agents”.

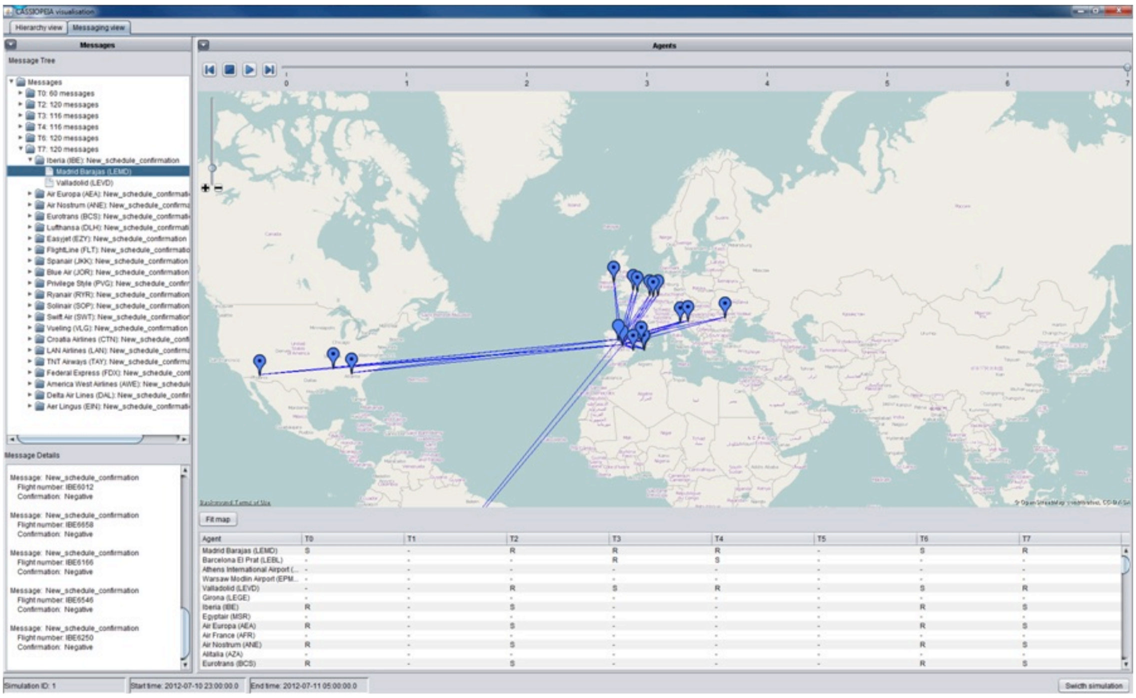

Figure 4.12: Messaging view

The section “messages” displays information about the messages sent by agents during the simulation. This section is divided into two parts:

- Message tree: shows structural information about all the messages,
- Message details: shows the content of the messages;

The message tree is a dynamic interactive tree that provides aggregated structural information about messages. Figure 4.13 shows the structure of the message tree and figure 4.14 shows a fragment of the structure. The time of the simulation is represented as a series of discrete time moments, represented as T0, T1, T2, etc. The message tree provides an aggregated view and groups the messages by time, sender and receivers. The tree shows a group of all the messages from sender X to receiver Y at a given time moment.

Level	Structure	Examples
1	Time and the total number of messages sent at this time.	T7: 120 messages.
2	Agents-senders and message topic.	Ryanair (RYR): New_schedule_confirmation
3	Agents-receivers.	Madrid Barajas (LEMD); Valladolid (LEVD).

Figure 4.13: Structure of message tree

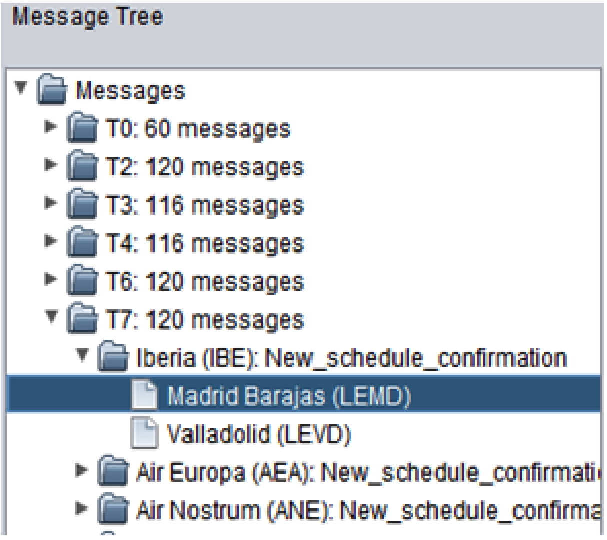

Figure 4.14: A fragment of message tree

The nodes of the tree are collapsed by default. The user can expand/collapse them by left-clicking. By left-clicking on the leaf of this tree (on the agent-receiver entity) the content of the message to the selected receiver from corresponding sender is shown in the area "message details". Figure 4.14 correspond to the case in which the user consults all the messages that were sent at *time moment 7* by *Iberia (IBE)* to *Madrid Barajas (LEMD)* with the topic of the message "*New_schedule_confirmation*". The user can find a list of messages in the "message details" area (Figure 4.15) corresponding to this interaction.

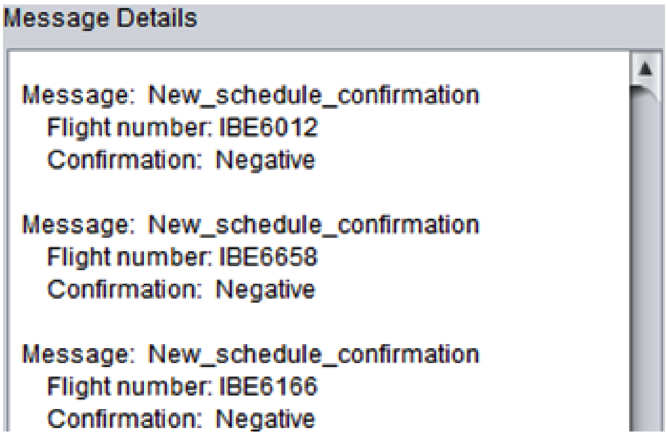

Figure 4.15: A fragment of "message details" area

The section "message details" contains media buttons and a slider to control other components of the view. The slider contains discrete time moments of the simulation time. At each time moment, the displayed information is updated in the message tree, map and agent activity table. There are 4 control buttons:

- "Step backward". Sets the position to the previous time moment. Updates the information on a map, according to the selected time moment. Message tree and activity table are not affected.
- "Stop". Sets the slider to the first time moment available.

- “Play/Pause”. In “Play” position makes slider move sequentially through all the discrete time moments without stop. In “Pause” position stays at the time moment at which it has been pressed.

- “Step Forward”. Sets the position of a slider to the next time moment. Updates the information of all the displaying components.

The geographic map shows a spatial view of the agent interaction. Each marker corresponds to an agent. Messages are represented as blue lines between senders and receivers. Senders are identified as markers with a black dot. The map includes zoom controls in the top left corner. Zoom level also may be controlled by the “fit map” button, which is situated right under the map. This button sets the appropriate zoom level to display all the markers of agents in a map.

The agent activity table includes columns for the time moments of the simulation. The rows correspond to the agents. Cells show the information about the type of activity an agent performs at each time moment (*S*: sends a message, *R*: receives a message).

The status pane (figure 4.17) shows context information and it is common for all the views of visualization windows. The status pane shows the ID of the simulation, start and end time of the simulation and it also contains a switch button that allows the user to change simulation ID.

Agent	T0	T1	T2
Madrid Barajas (LEMD)	S	-	R
Barcelona El Prat (LEBL)	-	-	-
Athens International Airport (...	-	-	-
Warsaw Modlin Airport (EPM...	-	-	-
Valladolid (LEVD)	-	-	R
Girona (LEGE)	-	-	-
Iberia (IBE)	R	-	S
Egyptair (MSR)	-	-	-
Air Europa (AEA)	R	-	S
Air France (AFR)	-	-	-
Air Nostrum (ANE)	R	-	S
Alitalia (AZA)	-	-	-
Eurotrans (BCS)	R	-	S

Figure 4.16: A fragment of agent activity table

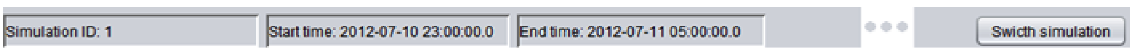

Figure 4.17: Status pane

4.3 Working memory user interface

The working memory contains initial values of the simulation together with intermediate and final generated results after the simulation execution. In order to consult the content of the working memory, the MySQL Workbench provides an appropriate user interface. To initiate the MySQL Workbench, the user creates a connection to the database (Figure 4.18) [MySQL AB, 2006].

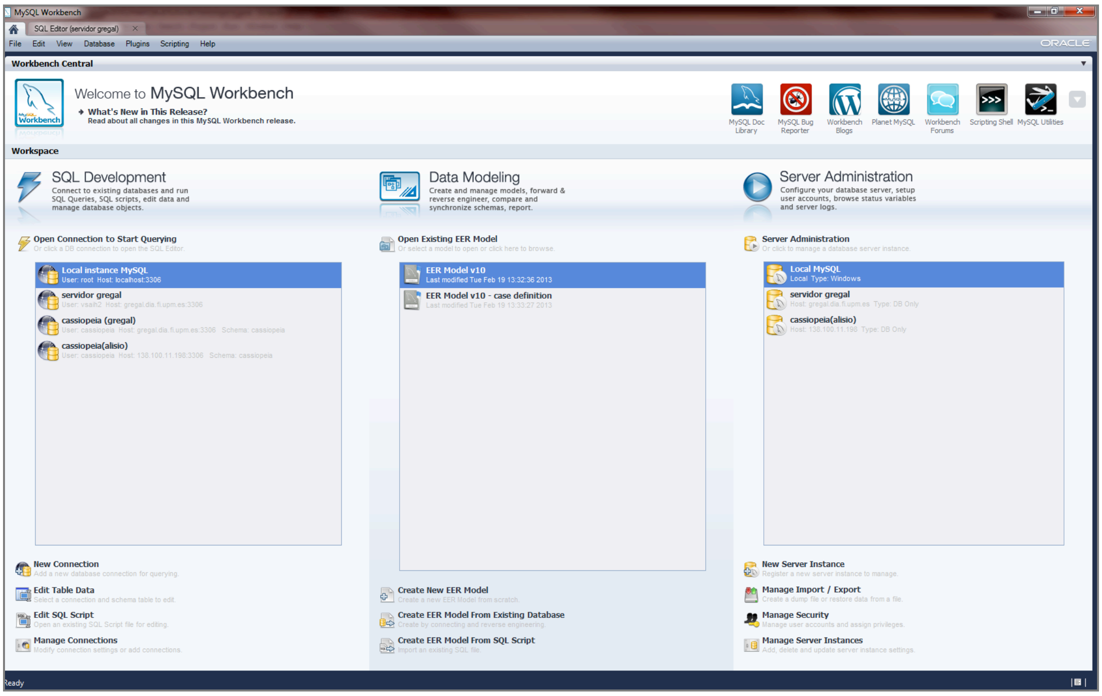

Figure 4.18: Example screen of the MySQL workbench

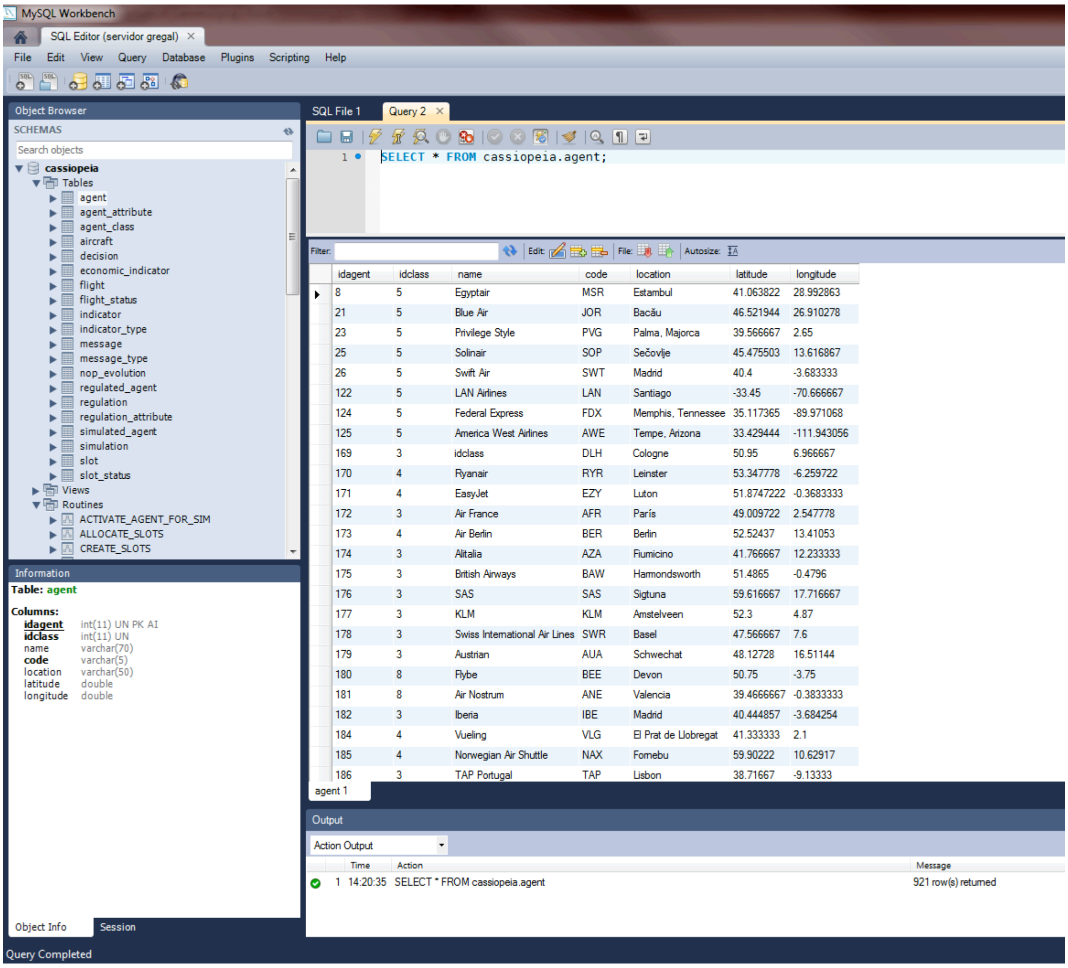

Figure 4.19: Example screen of the SQL Editor

When the connection to the database is established, the SQL Editor window is presented (figure 4.19). This window allows the user to consult and manipulate the content of the database. The SQL Editor includes four relevant regions:

- The *Object Browser* represents the database as a tree model categorized by tables, views (not used) and routines (procedures/functions).
- The *Information* region shows the columns of the selected table in the *Object Browser* (the information of the procedures is not relevant).
- The *Output* section shows a message to check if the query is running properly, the rows returned and the time used to retrieve the data. If the SQL query is wrong, the error is printed in this view.
- The *SQL File/Query* region is divided in two sections: the *SQL query* and the *Result Set*. The *SQL query* must be written according to the MySQL query syntax [MySQL AB, 2006].

An easy way to consult data is by right clicking in a table in the *Object Browser* and selecting *Select Rows* from the submenu. A SQL query is generated automatically. In the retrieved *Result Set* the data can be modified, deleted or added new rows. Any change in the database is confirmed using the *Apply* button. By default, MySQL Workbench limits the number of rows retrieved to 1,000. This limit can be changed/deleted in the Preferences of MySQL Workbench.

Figure 4.19 shows the data of some agents stored in the working memory. In particular, the KPI values generated by the simulation can be looked up in the table *Indicators*. Figure 4.20 shows an example of the window provided by the MySQL workbench to obtain the KPI values corresponding to different airports. To analyze this information, the user can order or filter this information using the utilities provided by the user interface. Figure 4.21 shows the status of the NOP for a simulation example (*simulation* 1) when the query is executed. This query calls the procedure *GET_CURRENT_NOP*.

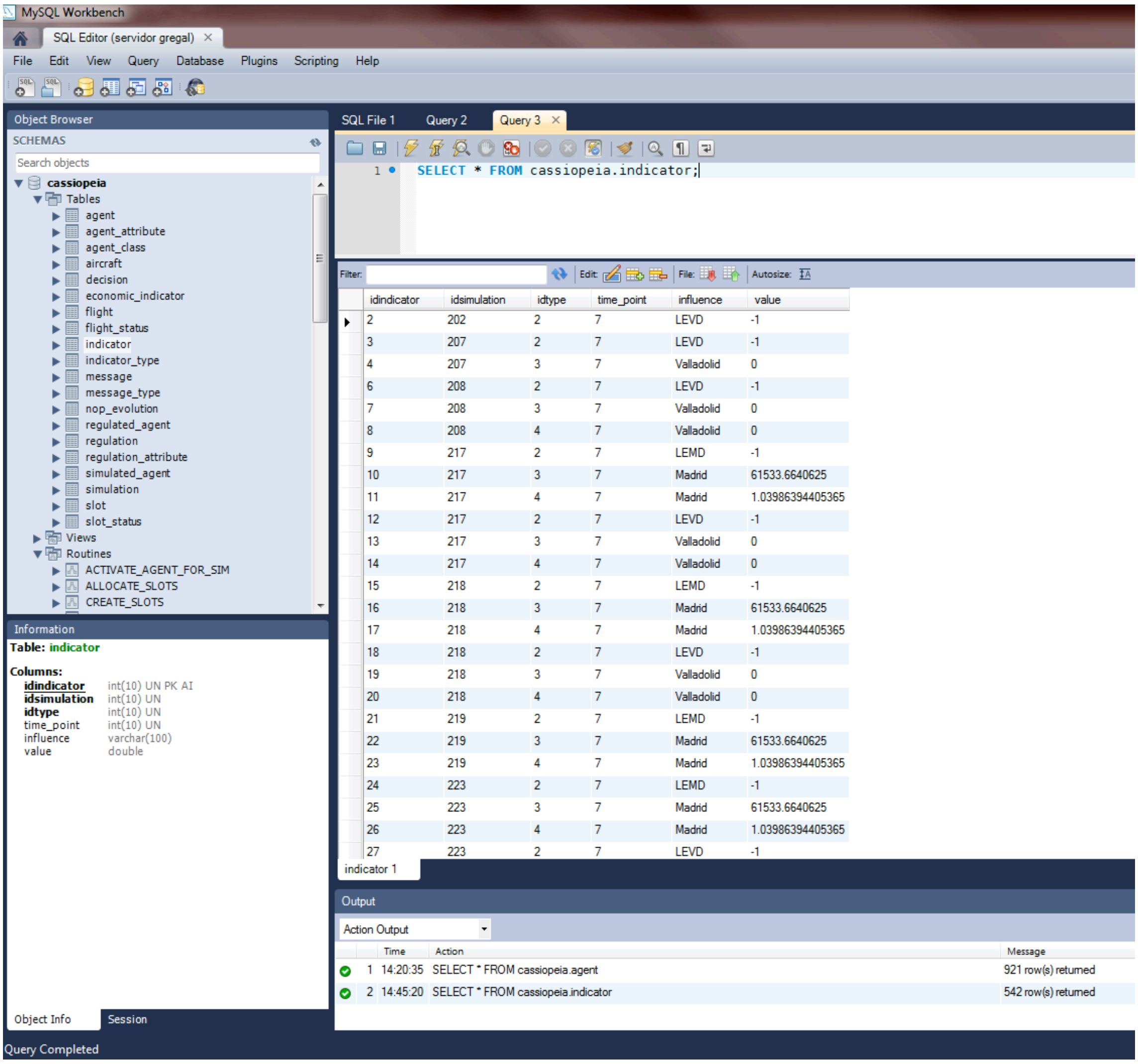

Figure 4.20: Example of KPI values generated after the simulation

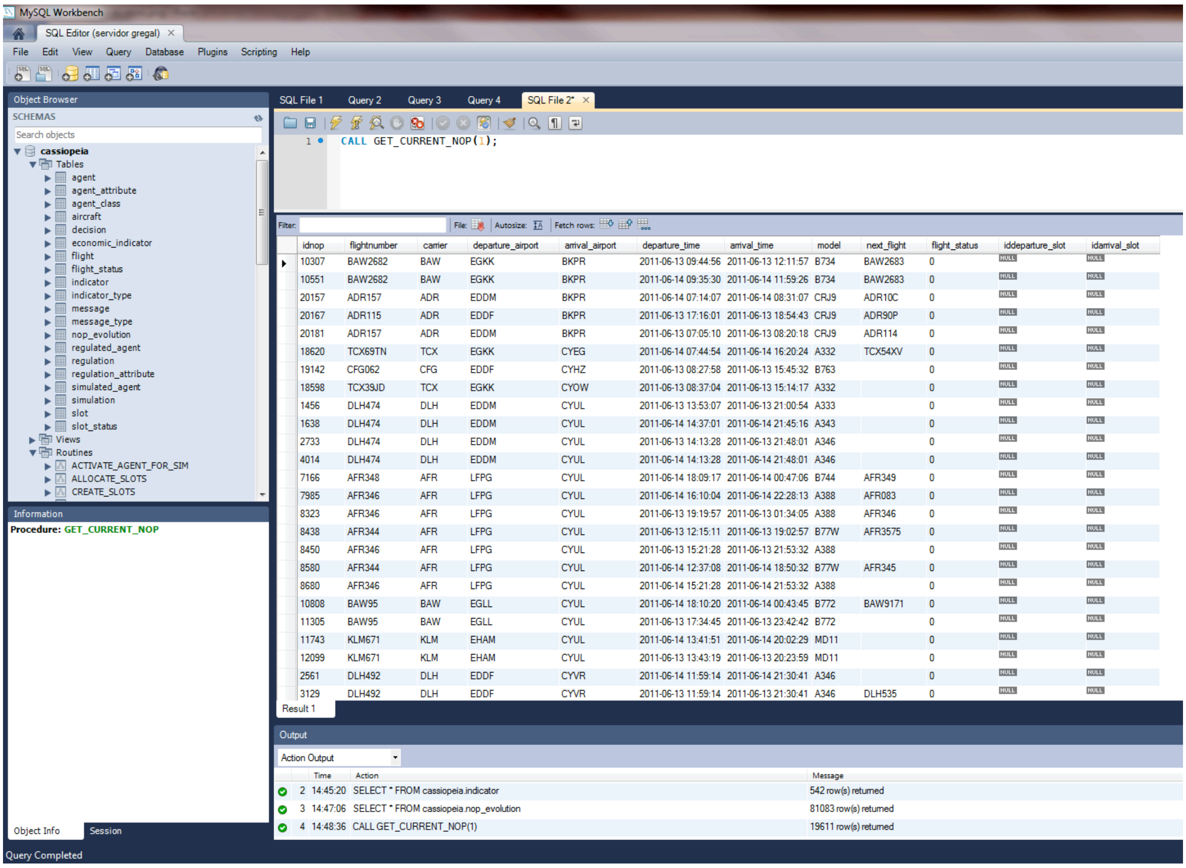

Figure 4.21: Example of NOP values generated during a simulation example

The MySQL Workbench also generates CSV output files from the contents of selected database tables. This is a convenient solution to use in combination with other software tools (e.g. MS Excel) that get the CSV files as input file and allow the user to manipulate the generated data and generate specific graphics to analyze the results. These files can be generated from the current *Result Set* shown by MySQL Workbench using the menu *Query>Export results*.

4.4 Simulation execution log

As a result of the simulation execution, a log text file is generated with a trace describing the linear sequence of steps performed during the execution of the simulation case. The execution log is useful for a detailed step by step analysis of the behavior of the simulation. This analysis can help the user understand the micro-level behavior of agents. It is also useful to help the user validate, refine or calibrate agent-based models. Figure 4.22 shows an example of a generated file. The logging messages include different content types with the following format:

- *Regulation*

  `Regulation: from <start-date> to <end-date> in <regulated_airports>`

- *Messages*

  `T<step>: <sender> -> <receiver> : <message_type>`
  `    <message_content>`

- *Decision*

  T<*step*>: <*agent*> : <*decision*>

- *Indicator*

  T<*step*>: <*agent*> : <*indicator*> : <*value*>

```
Regulation: from 23:00:00 to 05:00:00 in [LEMD].

T2: JKK -> LEVD : New schedule request
    Flight number:  JKK457
    Origin:         LEBL
    Departure time: 20:55:10
    Destination:    LEVD
    Arrival time:   22:01:10
    Duration(min):  66
    Aircraft type:  MD87

T1: LEMD -> AZA : Noncompliance
    Flight number: AZA91F

T3: LEVD -> MMMX : New schedule request
    Flight number:  IBE6401
    Requested Time: 10:31:39

T4: LEPA -> LEVD : Request approved
    Flight number: SWT102
    Time offered:  20:31:08

T5: LEVD -> SWT : New schedule reserved
    Flight number:   SWT102
    Departure time:  20:31:08
    Arrival time:    22:00:00

T4: GMMX -> LEVD : Request approved
    Flight number: EZY7898
    Time offered:  20:27:20

T4: LFML -> LEVD : Request approved
    Flight number: RYR5447
    Time offered:  20:37:15

T4: LIME -> LEVD : Request approved
    Flight number: RYR5996
    Time offered:  19:51:55

T4: LKPR -> LEVD : Request approved
    Flight number: CSA702
    Time offered:  18:48:34

T4: LEPA -> LEVD : Request approved
    Flight number: AEA6096
    Time offered:  20:54:32

T4: EGCC -> LEVD : Request approved
    Flight number: RYR58VN
```

Figure 4.22: Partial example of simulation execution log

5 Next steps and future deliverables

This document completes the development cycle of the Cassiopeia's software platform:

- D3.1 Software Requirements
- D3.4 System Implementation
- D3.6 System Evaluation

At this stage the software is ready to prepare, execute and analyze simulation cases. Therefore it provides input information for the deliverables related to development of Case Studies:

- D4.1: Study Report – Case Study 1
- D4.2: Study Report – Case Study 2
- D4.3: Study Report – Case Study 3

Note that these deliverables constitute the final pieces of the Cassiopeia project.

6 References

MySQL AB (2006): "MySQL Administrator's Guide and Language Reference, 2nd Edition". MySQL Press. Online Reference Manual: http://dev.mysql.com/doc/refman/5.5/en/

Pokahr, A., Braubach, L., Jander K. (2012): "The Jadex Project: Programming Model". In: Multiagent Systems and Applications. Web Site: http://activecomponents.org

Stott, J. (2006). Jcoord web site: http://www.jstott.me.uk/jcoord/

Zakhour, S. B., Kannan, S., Gallardo, R. (2013): "The Java Tutorial: A Short Course on the Basics (5th Edition) (Java Series)". Prentice Hall. API Website: http://docs.oracle.com/javase/7/docs/api/

Appendix A Example of definition of ATM network

This appendix describes an example of definition of ATM network. It is a simplified example for illustration purposes. The definition consists of two parts: (1) the definition of a case-specific agent model and (2) the definition of agent instances and agent-environment.

A.1 Agent model

This example of agent model includes three files[1]: (1) airport reschedule capability (file *reschedule.capability.xml*), (2) the manager agent (file *manager.agent.xml*) and (3) the application file (file *test01.application.xml*).

A.1.1 Airport reschedule capability

The next XML file is an example of a capability file for airport reschedule (file name *reschedule.capability.xml*). The reschedule capability of airports regulates several slots of the airport and assigns the affected flights to other slots interacting with the airlines. The file contains the header, a list of imports, beliefs, goals, plans, events and expressions, and finally the footer.

In this file, the reception of a *request_regulation* message activates the *apply_regulation* plan. It disables the slots during the regulation time and communicate to the airlines the flights affected using an *inform_noncompliance* message. Airports receive request_reschedule messages and prioritize that requests if the airport is regulated using *prioritize_request* plan (or using *not_prioritize_request* plan if they are not regulated). Airports assign their slots and ask for connecting slots for a suitable slot sending a *request_reschedule_conn* message. The slot assignation ends sending an *inform_reserved* message. Proposals can be accepted or refused, and airlines send accept_proposal or *refuse_proposal* messages to tell the decision to the airports. Those messages trigger the *confirmate_proposal* plan, modifying the NOP accordingly.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- <H3> Case 1: Airport reschedule capability.</H3> -->
<capability xmlns="http://jadex.sourceforge.net/jadex"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                      http://jadex.sourceforge.net/jadex-bdi-2.3.xsd"
  name="reschedule"
  package="simulator.airport">
<imports>
  <import>jadex.bridge.fipa.*</import>
  <import>simulator.manager.Regulation</import>
  <import>simulator.environment.FlightPlan</import>
  <import>simulator.airline.Request</import>
</imports>
<beliefs>
  <beliefref name="icao" >
    <abstract />
  </beliefref>
  <beliefref name="operations" >
    <abstract/>
  </beliefref>
  <beliefref name="simulation" >
    <abstract/>
  </beliefref>
  <beliefref name="round"  >
    <abstract />
  </beliefref>
```

[1] This example assumes that another agent class is defined (the airline).

```
    <belief name="regulated" class="boolean">
      <fact>false</fact>
    </belief>
    <beliefset name="requests"  class="Request" />
  </beliefs>
  <goals>
    <performgoal name="perform_assignment" >
      <parameter name="flight" class="FlightPlan" />
    </performgoal>
  </goals>
  <plans>
    <plan name="apply_regulation" >
      <body class="ApplyRegulationPlan"/>
      <trigger>
        <messageevent ref="request_regulation"/>
      </trigger>
    </plan>
    <plan name="prioritize_request">
      <body class="SchedulingPriorizationPlan"/>
      <trigger>
        <condition>$beliefbase.regulated</condition>
      </trigger>
      <waitqueue>
        <messageevent ref="request_reschedule"/>
      </waitqueue>
    </plan>
    <plan name="no_prioritize_request">
      <body class="SchedulingNoPriorizationPlan"/>
      <trigger>
        <messageevent ref="request_reschedule"/>
      </trigger>
      <precondition>!$beliefbase.regulated || round != 0</precondition>
    </plan>
    <plan name="assign_own_slot" >
      <body class="SlotAssignmentPlan"/>
      <trigger>
        <internalevent ref="perform_assignment" />
      </trigger>
    </plan>
    <plan name="assign_conn_slot">
      <body class="SuitableSlotPlan" />
      <trigger>
        <messageevent ref="request_reschedule_conn"/>
      </trigger>
    </plan>
    <plan name="confirmate_proposal">
      <body class="RescheduleConfirmationPlan"/>
      <trigger>
        <messageevent ref="accept_proposal"/>
        <messageevent ref="accept_proposal"/>
      </trigger>
    </plan>
  </plans>
  <events>
    <messageevent name="request_regulation" type="fipa">
      <parameter name="performative" class="String" direction="fixed">
        <value>SFipa.INFORM</value>
      </parameter>
      <match>$content instanceof Regulation</match>
    </messageevent>
    <messageevent name="inform_noncompliance" type="fipa">
      <parameter name="performative" class="String" direction="fixed">
        <value>SFipa.INFORM</value>
      </parameter>
    </messageevent>
    <messageevent name="request_reschedule" type="fipa">
```

```
      <parameter name="performative" class="String" direction="fixed">
        <value>SFipa.REQUEST</value>
      </parameter>
      <match>$content instanceof Request</match>
    </messageevent>
    <messageevent name="request_reschedule_conn" type="fipa">
      <parameter name="performative" class="String" direction="fixed">
        <value>SFipa.REQUEST</value>
      </parameter>
      <parameter name="reply_with" class="String">
        <value>SFipa.createUniqueId($scope.getAgentName())</value>
      </parameter>
      <match>$content instanceof FlightPlan</match>
    </messageevent>
    <messageevent name="inform_approved" type="fipa">
      <parameter name="performative" class="String" direction="fixed">
        <value>SFipa.AGREE</value>
      </parameter>
      <match>$content instanceof FlightPlan</match>
    </messageevent>
    <messageevent name="inform_reserved" type="fipa">
      <parameter name="performative" class="String" direction="fixed">
        <value>SFipa.INFORM</value>
      </parameter>
    </messageevent>
    <messageevent name="accept_proposal" type="fipa" direction="receive">
      <parameter name="performative" class="String" direction="fixed">
        <value>SFipa.ACCEPT_PROPOSAL</value>
      </parameter>
      <match>$content instanceof FlightPlan</match>
    </messageevent>
    <messageevent name="reject_proposal" type="fipa" direction="receive">
      <parameter name="performative" class="String" direction="fixed">
        <value>SFipa.REJECT_PROPOSAL</value>
      </parameter>
      <match>$content instanceof FlightPlan</match>
    </messageevent>
  </events>
  <expressions>
    <expression name="query_requests" >
      select Request $request
      from $beliefbase.requests
      order by $request.getPriority() desc
    </expression>
  </expressions>
  </capability>
```

A.1.2 Manager

The manager agent controls the execution and creates agent instances for the simulation. The next XML file is an example of the definition of the manager agent (file name *manager.agent.xml*). This example modifies the general definition of the manager agent (see the general definition in the deliverable D.3.4 System Implementation). It contains the header, a list of imports, capabilities, beliefs, goals, plans, events and configurations, and finally the footer.

The manager is configured to start the simulation when the agent is created. This plan, called start_simulation, creates the simulated agents using the cms_create_component goal defined on the cmscap capability. When those agents are created, it sends an inform_regulation message to the regulated airports and waits for the ending of regulation process.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- <H3Case 1: Manager Agent.</H3> -->
<agent xmlns="http://jadex.sourceforge.net/jadex"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
       xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                           http://jadex.sourceforge.net/jadex-bdi-
2.3.xsd"
       name="Manager"
       package="simulator.manager">
<imports>
  <import>jadex.bridge.fipa.*</import>
  <import>simulator.Regulation</import>
  <import>simulator.environment.Agent</import>
</imports>
<capabilities>
  <capability name="cmscap" file="jadex.bdi.planlib.cms.CMS"/>
  <capability name="dfcap"  file="jadex.bdi.planlib.df.DF" />
</capabilities>
<beliefs>
  <belief name="simulation" class="int" />
  <belief name="regulation" class="Regulation" />
</beliefs>
<goals>
  <!-- Used to start other agents. -->
  <achievegoalref name="cms_create_component">
    <concrete ref="cmscap.cms_create_component"/>
  </achievegoalref>
  <achievegoalref name="df_search" >
    <concrete ref="dfcap.df_search"/>
  </achievegoalref>
  <performgoal name="regulate_agents" />
</goals>
<plans>
  <plan name="start_simulation"  >
    <body class="StartSimulationPlan" />
  </plan>
  <plan name="regulate_agents" >
    <body class="RegulateAgentsPlan" />
    <trigger>
      <goal ref="regulate_agents"/>
    </trigger>
  </plan>
</plans>
<events>
  <!-- Message to inform airports about the application of a regulation.
-->
  <messageevent name="inform_regulation" direction="send" type="fipa">
    <parameter name="performative" class="String" direction="fixed">
      <value>SFipa.INFORM</value>
    </parameter>
    <match>$content instanceof Regulation</match>
  </messageevent>
</events>
<configurations>
  <configuration name="standard">
    <plans>
      <initialplan ref="start_simulation"  />
    </plans>
  </configuration>
</configurations>
</agent>
```

A.1.3 Application

The description of each specific simulation case contains an application file to describe the execution of the simulation. The next XML description shows an example of application file (file name *test01.application.xml*). It has three different component types, one for each agent used in the simulation case, so there are airports, airlines and managers, indicating its

description file location. The simulation is set for creating a manager component and controlling the simulation.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--  <H3>Validation Case 1 : Simulator</H3>  -->
<applicationtype xmlns="http://jadex.sourceforge.net/jadex"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                           http://jadex.sourceforge.net/jadex-
application-2.3.xsd"
       name="validateC1" package="simulator">

  <componenttypes>
    <componenttype name="Airport"
filename="simulator/airport/Airport.agent.xml"/>
    <componenttype name="Airline"
filename="simulator/airline/Airline.agent.xml"/>
    <componenttype name="Manager"
filename="simulator/manager/Manager.agent.xml"/>
  </componenttypes>

  <configurations>
    <configuration name="test">
      <components>
         <component type="Manager" name="manager"
configuration="standard" master="true"/>
       </components>
    </configuration>
  </configurations>
</applicationtype>
```

A.2 Agent instances and environment

This section illustrates with examples the definition of agent instances and environment. The section shows the content of database tables presented as it is shown by the MySQL Workbench. The definition of agent instances includes the following tables: agents (Figure A.1), attributes for agents (Figure A.2) and subclasses of agents (Figure A.3). The definition of the environment includes the following tables: flights (Figure A.4), flight status table with the allowable values for the state of a flight (Figure A.5) and aircrafts (Figure A.6).

idagent	idclass	name	code	location	latitude	longitude
1	3	Madrid Barajas	LEMD	Madrid	40.4500000	-3.5500000
2	3	Barcelona El Prat	LEBL	Barcelona	41.2969400	2.0783300
3	3	Athens International Airport	LGAV	Atenas	37.9363889	23.9444444
4	4	Warsaw Modlin Airport	EPMO	Nowy Dwór Mazowiecki	52.4511110	20.6516670
5	4	Valladolid	LEVD	Valladolid	41.7061120	-4.8519440
6	4	Girona	LEGE	Girona	41.9008330	2.7605560
7	5	Iberia	IBE	Madrid	40.4448570	-3.6842540
8	5	Egyptair	MSR	Estambul	41.0638220	28.9928630
9	5	Air Europa	AEA	Mallorca	39.4833330	2.8511110
10	5	Air France	AFR	París	49.0097220	2.5477780
11	5	Air Nostrum	ANE	Valencia	39.4666667	-0.3833333
12	5	Alitalia	AZA	Fiumicino	41.7666670	12.2333330
13	5	Eurotrans	BCS	Zaventem	50.8867920	4.4748120
14	5	British Airways	BAW	Harmondsworth	51.4865000	-0.4796000
15	5	Brussels Airlines	BEL	Diegem	50.8942320	4.4356230
16	5	Czech Airlines	CSA	Prague	50.0833330	14.4166670
17	5	Lufthansa	DLH	Cologne	50.9500000	6.9666670
18	6	Easyjet	EZY	Luton	51.8747222	-0.3683333
19	5	FlightLine	FLT	Rochford	51.5821000	0.7065000
20	5	Spanair	JKK	L'Hospitalet de Llobre...	41.3588890	2.0994170
21	5	Blue Air	JOR	Bacău	46.5219440	26.9102780
22	5	KLM Royal Dutch Airline	KLM	Amstelveen	52.3000000	4.8700000
23	5	Privilege Style	PVG	Palma, Majorca	39.5666670	2.6500000
24	6	Ryanair	RYR	Leinster	53.3477780	-6.2597220
25	5	Solinair	SOP	Sečovlje	45.4755030	13.6168670
26	5	Swift Air	SWT	Madrid	40.4000000	-3.6833330

Figure A.1: Example content of the agent table

idagent_attr	idagent	attribute	value
1	1	opening	2011-08-01 20:00:00
2	1	closing	2011-08-02 08:59:59
3	1	operations	80
8	2	opening	2011-08-01 20:00:00
9	2	closing	2011-08-02 08:59:59
10	2	operations	80
15	3	opening	2011-08-01 20:00:00
16	3	closing	2011-08-02 08:59:59
17	3	operations	80
22	4	opening	2011-08-01 20:00:00
23	4	closing	2011-08-02 08:59:59
57	4	operations	80
24	5	opening	2011-08-01 21:00:00
25	5	closing	2011-08-02 07:00:00
26	5	operations	800
28	6	opening	2011-08-01 21:00:00
29	6	closing	2011-08-02 07:00:00
30	6	operations	300
32	7	homebase	LEMD
172	7	homebase	LEBL
33	8	homebase	GECA
34	9	homebase	LEBL
35	10	homebase	LFPG
36	11	homebase	LEVC
37	12	homebase	LIML
38	13	homebase	EGLW

agent_attribute 1

Figure A.2: Example content of the agent_attribute table

idclass	classname	idparentclass
1	Airport	NULL
2	Airline	NULL
3	Regulated_Airport	1
4	Non_regulated_airport	1
5	Network_airline	2
6	Low cost	2
NULL	NULL	NULL

Figure A.3: Example content of the agent_class table

flightnumber	idcarrier	departure_time	arrival_time	iddeparture_airport	idarrival_airport	idaircraft	status	next_flight	min_turnaround_time	previous_flight
LAN705	122	2011-08-01 23:00:00	2011-08-02 11:22:12	1	86	7	0	LAN705B	00:00:50	LAN705A
EZY7789	18	2011-08-01 23:00:00	2011-08-01 23:49:48	1	87	3	0	EZY7789B	00:00:50	EZY7789A
ANE8984	11	2011-08-01 23:19:48	2011-08-01 23:49:48	1	88	15	0	ANE8984B	00:00:50	ANE8984A
IBE6843	7	2011-08-01 23:30:00	2011-08-02 10:57:00	1	89	8	0	IBE6843B	00:00:50	IBE6843A
IBE6825	7	2011-08-01 23:40:12	2011-08-02 09:39:00	1	90	8	0	IBE6825B	00:00:50	IBE6825A
IBE6831	7	2011-08-01 23:45:00	2011-08-02 12:01:48	1	86	7	0	IBE6831B	00:00:50	IBE6831A
IBE6659	7	2011-08-01 23:45:00	2011-08-02 10:46:12	1	99	7	0	IBE6659B	00:00:50	IBE6659A
IBE6849	7	2011-08-01 23:49:48	2011-08-02 09:36:00	1	91	7	0	IBE6849B	00:00:50	IBE6849A
IBE0564	7	2011-08-02 00:10:12	2011-08-02 00:58:48	1	92	4	0	IBE0564B	00:00:50	IBE0564A
BCS8463	13	2011-08-02 00:28:48	2011-08-02 03:10:12	1	93	2	0	BCS8463B	00:00:50	BCS8463A
IBE6013	7	2011-08-02 00:40:12	2011-08-02 11:58:12	1	94	7	0	IBE6013B	00:00:50	IBE6013A
PVG7995	23	2011-08-02 01:04:48	2011-08-02 06:16:12	1	95	12	0	PVG7995B	00:00:50	PVG7995A
FTL702	19	2011-08-02 01:34:48	2011-08-02 03:09:00	1	2	17	0	FTL702B	00:00:50	FTL702A
JKK3211	20	2011-08-02 01:45:00	2011-08-02 06:12:00	1	96	4	0	JKK3211B	00:00:50	JKK3211A
IBE6401	7	2011-08-02 02:10:12	2011-08-02 12:49:12	1	97	7	0	IBE6401B	00:00:50	IBE6401A
FTL851	19	2011-08-02 02:34:48	2011-08-02 03:58:48	1	87	17	0	FTL851B	00:00:50	FTL851A
SWT182	26	2011-08-02 03:00:00	2011-08-02 05:22:12	1	98	9	0	SWT182B	00:00:50	SWT182A
SOP9013	25	2011-08-02 03:45:00	2011-08-02 06:10:12	1	98	10	0	SOP9013B	00:00:50	SOP9013A
RYR5991	24	2011-08-02 04:19:48	2011-08-02 06:06:00	1	100	11	0	RYR5991B	00:00:50	RYR5991A
DLH32J	17	2011-08-02 04:34:48	2011-08-02 06:43:48	1	101	5	0	DLH32JB	00:00:50	DLH32JA
RYR2A	24	2011-08-02 04:40:12	2011-08-02 07:09:00	1	102	11	0	RYR2AB	00:00:50	RYR2AA
RYR9674	24	2011-08-02 04:45:00	2011-08-02 06:34:48	1	103	11	0	RYR9674B	00:00:50	RYR9674A
EZY783F	18	2011-08-02 04:49:48	2011-08-02 08:07:48	1	104	3	0	EZY783FB	00:00:50	EZY783FA
RYR5456	24	2011-08-02 04:49:48	2011-08-02 05:25:12	1	105	11	0	RYR5456B	00:00:50	RYR5456A
VLG3418	27	2011-08-01 22:30:00	2011-08-01 23:13:12	87	1	4	0	VLG3418B	00:00:50	VLG3418A
AEA6096	9	2011-08-01 22:25:12	2011-08-01 23:13:48	106	1	11	0	AEA6096B	00:00:50	AEA6096A

Figure A.4: Example content of the flights table

Filter:

idstatus	statusname
0	confirmed
1	noncompliant
2	reserved
3	cancel

Figure A.5: Example content of the flight_status table

Filter:

idaircraft	model
1	A306
2	A30B
3	A319
4	A320
5	A321
6	A332
7	A343
8	A346
9	B733
10	B734
11	B738
12	B752
13	B762
14	B764
15	CRJ2
16	MD83
17	SW4

Figure A.6: Example content of the aircrafts table